Second lab: Hazelcast

Author: Khrystyna Kokolus
Github: https://github.com/khristinakokolus/microservices_architecture.git

All code is in the *hazelcast_basics* branch.

Task 1: **Demonstration of work of Distributed map**

Created Hazelcast client, distributed map and added 1000 values there.

```
D:\UCU\Third\microservices_architecture\hazelcast>python3 distributed_map.py
Starting connecting to the Hazelcast client...
Connected to the Hazelcast client
Created a distributed map
1 value
2 value
3 value
4 value
5 value
6 value
7 value
8 value
9 value
10 value
11 value
```

```
986 value
987 value
988 value
989 value
990 value
991 value
992 value
993 value
994 value
995 value
996 value
997 value
998 value
999 value
1000 value
Shutdown Hazelcast client
```

Initially distributed data on three nodes.

Map Statistics (In-Memory Format: BINARY)

Member	Entries	Gets	Puts	Removals	Sets	Entry Me...
10.10.225.57...	347	0	346	0	0	45.03 kB
10.10.225.57...	343	0	344	0	0	44.51 kB
10.10.225.57...	310	0	310	0	0	40.24 kB
TOTAL	1000	0	1000	0	0	129.78 kB

When deleted one node:

Map Statistics (In-Memory Format: BINARY)

Member	Entries	Gets	Puts
10.10.225.57...	494	0	346
10.10.225.57...	506	0	344
TOTAL	1000	0	690

When deleted two nodes.

Map Statistics (In-Memory Format: BINARY)

Member	Entries	Gets	Puts
10.10.225.57...	1 000	0	346
TOTAL	1000	0	346

So, the data is stored regardless of the number of nodes

Task 2: **Demonstration of work of Distributed Map with locks**

```
D:\UCU\Third\microservices_architecture\hazelcast>python3 distributed_map_with_locks.py
Starting connecting to the Hazelcast client for optimistic update...
Starting connecting to the Hazelcast client for racy update...
Connected to the Hazelcast client for optimistic update
Connected to the Hazelcast client for racy update
Created a distributed map for optimistic update
Created a distributed map for racy update
Starting optimistic update...
Optimistic. At:  0
Starting racy update...
Racy. At:  0
Starting connecting to the Hazelcast client for pessimistic update...
Optimistic. At:  10
Optimistic. At:  20
Connected to the Hazelcast client for pessimistic update
Optimistic. At:  30
Created a distributed map for pessimistic update
Starting pessimistic update..
Racy. At:  100
Racy. At:  200
Racy. At:  300
Racy. At:  400
Racy. At:  500
Racy. At:  600
Racy. At:  700
```

```
Racy. At:  700
Racy. At:  800
Racy. At:  900
Finished racy update! Result =  1009
Finished for pessimistic update! Result =  1081
```

```
Optimistic. At:  890
Optimistic. At:  900
Optimistic. At:  910
Optimistic. At:  920
Optimistic. At:  930
Optimistic. At:  940
Optimistic. At:  950
Optimistic. At:  960
Optimistic. At:  970
Optimistic. At:  980
Optimistic. At:  990
Finished for optimistic update! Result =  2042
```

Task 3: **Bounded Queue**

In my case queue has max size of 10
One client is a **producer** and two other are **consumers**

```
D:\UCU\Third\microservices_architecture\hazelcast>python3 bounded_queue_producer_consumer.py
Starting connecting to the Hazelcast client for consumer...
Starting connecting to the Hazelcast client for consumer...
Connected to the Hazelcast client for consumer
Starting connecting to the Hazelcast client for producer...
Created a distributed queue for consumer
Connected to the Hazelcast client for consumer
Created a distributed queue for consumer
Connected to the Hazelcast client for producer
Created a distributed queue for producer
Producing:  0
Consumed:  0 value . Process:  2
Producing:  1
Consumed:  1 value . Process:  3
Producing:  2
Consumed:  2 value . Process:  3
Consumed:  3 value . Process:  2
Producing:  3
Producing:  4
Consumed:  4 value . Process:  3
Producing:  5
Consumed:  5 value . Process:  2
Producing:  6
Consumed:  6 value . Process:  3
Producing:  7
Consumed:  7 value . Process:  2
Producing:  8
Consumed:  8 value . Process:  3
Producing:  9
Consumed:  9 value . Process:  3
Producing:  10
```

From here can be seen that consumers receive data mostly in turn.

Here is shown what happens when the Bounded Queue is full and there are no consumers. It hangs.